

# Personal Marks and Community Certificates: Detecting Clones in Mobile Wireless Networks of Smart-Phones

Marco Valerio Barbera  
Department of Computer Science  
Sapienza University of Rome

Alessandro Mei\*  
Dept of Computer Science  
Sapienza University of Rome, and  
Dept of Computer Science and Engineering  
University of California San Diego, USA

## ABSTRACT

We consider the problem of detecting clones in wireless mobile ad-hoc networks. We assume that one of the devices of the network has been cloned. Everything, including certificates and secret keys. This can happen quite easily, because of a virus that immediately after sending all the content of the infected device to the adversary destroys itself, or just because the owner has left his device unattended for a few minutes in a hostile environment. The problem is to detect this attack. We propose a solution in networks of mobile devices carried by individuals. These networks are composed by nodes that have the capability of using short-range communication technology like blue-tooth or Wi-Fi, where nodes are carried by mobile users, and where links appear and disappear according to the social relationships between the users. Our idea is to use social physical contacts, securely collected by wireless personal smart-phones, as a biometric way to authenticate the legitimate owner of the device and detect the clone attack. We introduce two mechanisms: Personal Marks and Community Certificates. Personal Marks is a simple cryptographic protocol that works very well when the adversary is an insider, a malicious node in the network that is part, or not very far, from the social community of the original device that has been cloned. Community Certificates work very well when the adversary is an outsider, a node that has the goal of using the stolen credentials when interacting with other nodes that are far in the social network from the original device. When combined, these mechanisms provide an excellent protection against this very strong attack. We prove our ideas and solutions with extensive simulations in a real world scenario—with mobility traces collected in a real life experiment.

\*This work was performed while Alessandro Mei was a Marie Curie Fellow at the Department of Computer Science and Engineering, University of California San Diego, USA. The fellowship is funded by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 253461.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.  
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

## Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—complexity measures, performance measures

## General Terms

Security

## Keywords

Delay tolerant networks, pocket switched networks, clone detection, community authentication.

## 1. INTRODUCTION

You have left your smartphone on the table at a cafeteria. You soon realize and go back to take it. Fortunately, it is still there! You feel safe—while you are not safe at all. An adversary has connected your smartphone to a laptop and dumped all of its memory, including public and secret cryptographic keys. It is a matter of seconds or, at most, minutes. You don't revoke your certificates (you feel safe!) and, a month later, you discover that your credentials have been used by someone else. If you think this cannot happen—people take very good care of their personal devices—consider that according to a fairly recent report (WTOP, 15 Nov 2006) 478 laptops have been lost or stolen from the IRS (the Internal Revenue Service is the United States federal government agency that collects taxes and enforces the internal revenue laws) between 2002–2006; 112 held sensitive taxpayer data, including SSNs.

Portable personal devices—smart-phones, laptops, and PDAs—are more and more used in our everyday life. We use them to make phone calls, to plan our activities, to surf the web, to manage our banking account, and probably very soon to make purchases. The system we consider is a network of personal smart-phones that connects to each other by using short-range communication technology like blue-tooth. The nodes are the devices and the links appear and disappear as people move and get in physical touch. As a consequence, the network is not a collection of randomly moving objects—it has a social structure that can be exploited to deliver revolutionary applications, and, we believe, new and surprisingly effective solutions to classical networking and security problems.

These networks have been called with several different names—*pocket switched networks* [20, 8], *mobile social networks*, *opportunistic mobile ad-hoc networks*, among others—and have drawn the attention of many researchers in the community. Most of the work has focused on forwarding, that is how to route messages in

such a way to deliver them as quickly as possible and as cheaply as possible to destination. Here, we focus on security. The attack that we have described in the first rows of this paper is the *clone attack*. The clone attack might as well be performed by a virus that infects our device and sends all the data, including secret keys associated with the certificates, to the adversary. So, you are not safe even though you never forget your smart-phone at the coffee shop. In this paper we are looking for a solution to this problem. Of course, we do not want one that consists in typing a password every time we use our certificates. People don't like passwords and tend to forget them, or, even worse, choose trivial ones.

Our idea is to use social physical contacts, securely collected by wireless personal smart-phones, as a biometric way to authenticate the owner of the device. Indeed, our social physical contacts—our family members, the same barman everyday at the coffee shop, our colleagues at work, our old friends when we hang out and relax after work—characterize ourselves in a distinctive way. Of course, each day is somewhat different and we don't always meet the same people, but surely there is a strong regularity in the communities we live in and in circle of friends that we usually meet in person. We will see how to use this regularity, complemented with other essential mechanisms, to detect the clone attack in wireless networks of smart-phones or other personal devices and to prevent the misuse of stolen certificates.

In this paper we introduce two protocols: Personal Marks and Community Certificates. Personal Marks is a simple cryptographic protocol that proves to be an excellent way to detect the attack when the adversary is an *insider*, a person that belongs to the same community of the victim. Community Certificate is a solution based on certificates that tells how the node is expected to behave in terms of social contacts. If the clone is an *outsider*, a node that behaves in a different way with respect to the victim and meets different nodes and different communities, then Community Certificate works very well, the certificate soon expires, and the clone cannot authenticate any more to the other nodes of the network. The two protocols are meant to be used at the same time and, according to a large set of experiments made with well-known traces of human contacts computed during real life experiments, they collectively prove to protect the nodes against the clone attack in an excellent way.

## 2. RELATED WORK

The detection of the clone attack is one of the most investigated security issue in wireless ad-hoc networks. As far as static wireless networks are concerned, there are three main approaches to the problem: Centralized, local and distributed random based techniques. The centralized techniques [19, 4] require nodes to send either location information or key information along with the node ID to a base station which in turn processes the information received and, in case of anomalies (same node ID with different locations), trigger a revocation procedure. Aside from presenting a single point of failure (the base station), this approach has the drawback of incurring high communication cost to the network due to the high number of messages involved.

Local-based schemes [9, 16, 19, 23] make use of voting mechanisms within nodes' neighborhoods to detect clones. Though they don't suffer from the drawbacks of the central based schemes, they fail on detecting replication attacks where node clones are scattered in different areas of the network.

The distributed and random based techniques [35, 14, 13, 40] require nodes to send signed location information to randomly selected destinations on the network in a hop-by-hop fashion. These schemes rely on the high probability of intersection of different

location-declaration routing paths started from replicas of the same node. The nodes on which such intersection occurs are called witnesses, and their task is to then trigger node revocation procedures. Another distributed scheme [11] uses a random value, distributed to the nodes by the base station, to generate independent clusters in the network. Each cluster is an Exclusive Subset Maximal Independent Set (ESMIS) and has a cluster head denoted with SLDRs. One or more trees (whose nodes correspond to SLDRs) are defined over the network graph, and bottom-up aggregation protocols on such trees are used to detect node replicas (that are present in more than one ESMIS).

Although these distributed techniques do not present single points of failure, the overhead they incur either in message traffic or in computational terms [11] is far from negligible. Moreover, all the aforementioned techniques, by relying on fixed geographical position of the nodes in the network are not apt to be used in mobile scenarios such the one we consider [20, 8].

With the increasing use of cellular technology, cloning of mobile devices for the purpose of making fraudulent telephone calls become a real threat to phone carriers. While this was a real threat in the nineties when CDMA was adopted [36], in the GSM era it started to be considered as a minor/non feasible attack, at least for fraudulent billing purposes. With the spreading of the mobile pay systems (72.8 million of users at the end of 2009, expected 220 million by 2011 in China only [33]), and especially those based on credit card transactions, phone cloning has become a vicious threat to the users portfolio [37]. Thus a lot of research on detecting such attacks have been done, most of which is based on the use of neural networks by the carrier to detect possible anomalies. For a good survey see [10].

The idea of exploiting information regarding social ties between nodes is not new, actually it is common to a good part of the literature on pocket switched networks (PSN) and similar social networks. Much research has been dedicated to the analysis of the data collected during real-life experiments, to compute statistical properties of human mobility, and to uncover its structure in sub-communities [38, 20, 8, 24, 7, 5, 6]. Later on, most of the work in the field focused on message forwarding and to find the best strategy to relay messages in order to route them to destination as fast as possible (see [15, 21, 18], among many others). Also security problematics such as node capture [12] or selfishness [28, 32] have been solved by making use of social relationships among nodes.

Biometrics have been used for centuries to authenticate people. The most common example of biometrics is the handwritten signature that we use everyday to certify our identity. With technological development, the use of biometrics to authenticate has become a well-known area of investigation in computer security. Several biological measurements have been used to identify people to computer systems: voice, face, iris, keystroke dynamics (the way we type is often sufficiently unique to identify ourselves), and others. In many cases these are not fully automatic. With keystroke dynamics, for example, the user has to type. As an excellent reference and starting point on computer biometrics, see [2].

The authors in [1] propose two intrusion detection systems that have similar biometrics ideas: The first one is build upon Radio Frequency Fingerprinting (RFF), whereas the second one leverages User Mobility Profiles (UMP). However both detection systems are centralized, and rely on the fact that the intruder (the clone) behaves substantially differently from the real user in terms of geographical movements. Thus, compared with the solutions proposed in this paper, both systems are based on a completely different idea and are not able to detect anomalies when the clone behaves similarly to the original node (for example, when the clone attack happens in

a building). Lastly, the solutions are not distributed.

To the best of our knowledge, this is the first paper that presents biometric authentication techniques based on the social contacts of the owner or the device to detect the clone attack in mobile wireless networks of smart-phones. The techniques, namely *Community Certificates* and *Personal Marks*, are *only* build upon the social-guided meeting patterns of users in the mobile social network. They are thoroughly orthogonal to trust or voting mechanisms, or geography based techniques to detect cellphone theft or credit card fraud, that are based on different ideas and can be used at the same time.

### 3. THE SYSTEM

Our network setting is made of last generation smart-phones. These devices are more and more popular—we can easily envision a society in which almost everybody will carry a personal computing device of this kind. Commercial smart-phones are usually able to communicate by using short-range communication technology, like blue-tooth and/or Wi-Fi, aside classic GSM. When two people meet, their personal devices establish a link by using one of the above mentioned short-range technologies. In the network we consider, nodes are devices carried by people, and links appear and disappear as people move and meet.

Smart-phones are not-so-small devices that can easily handle video/audio streaming, 3D games, web surfing and SSL sessions, and other applications. Therefore, we can safely assume that nodes are able to perform public key cryptography that is used to sign messages and to establish secure communication sessions among peers. The nodes are equipped with public/private key pairs, and the former is signed by a trusted authority CA. We assume that our protocols execute by setting up authenticated and encrypted sessions among the peers. Nodes are loosely time synchronized. Loose time synchronization is very easy to get, if a precision in the order of the second is enough, like in our protocols. We also assume that the trusted authority is able to send a message to any node in the system, for example using the cellular network. When a clone is present, the message is received by the original node and by the clone as well (of course it is perfectly possible that the clone has turned off its interface to the cellular network).

Our protocols rely on authenticated logs of physical contacts between devices. However, if not carefully implemented, these logs can be attacked. We assume that the nodes use techniques like distance bounding [3, 39] to guarantee that the log of a contact between two devices can be collected only when the devices are provably in physical proximity. These mechanisms are based on measuring the delay in the communication.

Lastly, we assume that the users have access to an alternative way to authenticate to the authority. There are several examples of such mechanisms. One example is Gmail: If you forget your password, you can still authenticate by responding to a list of personal questions that, most probably, only you can respond. In other systems, you might be able to authenticate by using a smart-card at your desktop at home. Another example is the pair of PIN and PUK codes used in cellular phones, the first one is short and easy to type, the second one longer, more tedious, and is meant to be used only to recover from PIN blocking or other uncommon events. In any case, we assume that the alternative mechanism to authenticate is secure but long, burdensome, and we definitely want to use it only in rare and exceptional circumstances like when we need to recover from a clone attack.

In the rest of this paper, we use  $\langle m \rangle_i$  to denote a message  $m$  signed by node  $i$ .

### 3.1 The Adversary and the Problem

We consider the following scenario: A smart-phone is infected by a worm or by a virus. The goal of the virus is to make a perfect copy of the device of the victim, to send all the data to the adversary's device, and to destroy itself without leaving any trace. A similar attack can be performed in a different way: You left your device unattended for just a couple of minutes; when you go back to take it, it is still there. What you don't know is that the adversary has made a clone by copying all the memory of your device.

Later on, the clone can be used to interact in the network by using the victim's credentials. Note that everything has been cloned, including certificates and secret keys. We assume that the clone can be switched off arbitrarily by the adversary to prevent detection—the device might be turned on only when necessary (that is, when it is used to connect to the other peers in the network with the stolen credentials). To start with, we assume that the adversary has been able to clone only one device. Later in the paper we will discuss what is going to happen if more devices can be stolen.

Our goal is to build a system able to detect and/or stop as fast as possible and as accurately as possible this attack.

### 4. DETECTION OF THE CLONE ATTACK

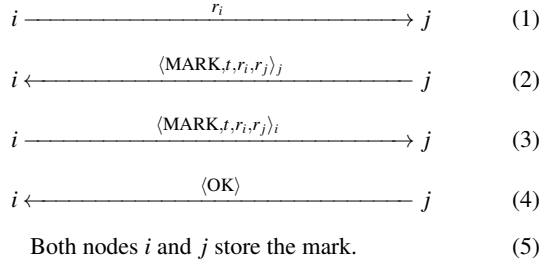
The users of our system are humans—their mobility pattern, the people they meet, and the places they visit are not random. Rather, the dynamics of the network is highly influenced by the fact that almost everything in our everyday life is guided by social relationships (e.g. friends, family) and interests (e.g. work, school, gym, or playing chess). If we think of our daytime, it usually follows a do-repeat cycle that might look like this: Go to work (or school); meet with colleagues; finish work and go out with the same old friends or do whatever we like to do during our free time; go back home; go to work again and so on. Of course, it is not always like that (fortunately), but surely there is a regularity in the things we do and the people we meet. The same cycle is performed by our mobile device, and the ones of our friends, family members, or colleagues that we meet over and over again. The people (devices) we meet everyday characterize ourselves in a remarkably distinctive way. One of the ideas in this paper is to use this characterization as efficiently as possible in order to make nodes prove who they are not with something they know (passwords), or something they have (certificate), but with the proof that they actually meet physically and regularly the people they are supposed to meet.

#### 4.1 Personal Marks

We start by introducing Personal Marks. Personal Marks is a very simple cryptographic protocol we can use to distinguish clones. Actually, Personal Marks is able to tell you that the person that you have before is not the same person that you met last time, though it can't really tell who is the impostor.

Let's make a step back and proceed more formally. Personal Marks consists in *marking* each node in your community with a small cryptographic object, signed by yourself, that you can check to have some level of guarantee that node  $j$  is exactly the same node  $j$  you have met earlier. First of all, we call *community* of node  $i$  its set of friends—the set of other nodes that gets frequently in physical contact with node  $i$ . One way to compute the community of node  $i$  is by using the distributed community detection algorithm described in [22]. Communities are a fundamental notion in social networks, people meet more often other people of the same communities and this intuitive property have been leveraged in the literature in several ways.

Personal Marks works as follows: When two nodes of the same



**Figure 1: Mark update protocol. Here we assume  $ID_i < ID_j$ .**

community  $i$  and  $j$  meet, they both put a mark on the other device. The mark that node  $i$  gives to node  $j$  contains a timestamp signed by node  $i$  itself. Same is done by node  $j$  and given to node  $i$ . The next time they meet, they both request the mark to check it before starting the new session.

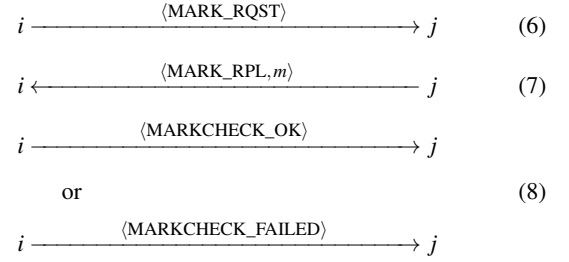
If there is a clone present in the community, say a clone  $j'$  of original node  $j$ , and  $j'$  gets in touch with the community of  $j$ , Personal Marks is going to detect the anomaly.

Assume that node  $j$  has been cloned at time  $t_0$  and that node  $i$  gets in physical contact with node  $j$  for the first time after the attack at time  $t_1 > t_0$ . Clearly, node  $i$  does not know if the node is the original or the clone and, by using Personal Marks, it freshens the mark with the node. The detection is triggered when node  $i$  meets the *other* node, that cannot help but exchanging the old version of the mark, a mark with a timestamp in the interval  $(-\infty, t_0)$  that does not pass the test by node  $i$ . If the adversary is an insider, node  $i$  will soon meet both the nodes, the clone and the original one (that is a friend). While node  $i$  is not able to tell who is the original node and who is the impostor, the clone attack is detected and node  $j$  is revoked by the authority. The legitimate node  $j$  has now to go through an alternative, longer procedure to re-authenticate.

#### 4.1.1 Personal Marks: The Check and the Update Protocols

In our system, a personal mark is a small cryptographic object that carries some randomness from both peers. The authentication protocol in Personal Marks is made of two parts: The mark-check, and the mark-exchange. Let us start from the mark-exchange protocol. Each time two nodes  $i$  and  $j$  meet, they contribute some randomness  $r_i$  and  $r_j$  to freshen the mark (or to establish one if this is the first time the nodes meet). The mark is of the form  $\langle \text{MARK}, t, r_i, r_j \rangle$ , where  $t$  is a timestamp (the timestamp is not crucial for the system to work, but it gives information about the timing of the attack in case of detection). The mark is signed and stored by both peers. Note that, if the protocol is not completed (may be for lack of continuous connectivity), the peers can safely assume that it has not happened.

Let us now look at the mark-check protocol, executed before the mark-exchange protocol when it is not the first time that nodes  $i$  and  $j$  meet. Suppose, without loss of generality, that first node  $i$  checks node  $j$  and then node  $j$  checks node  $i$ . Node  $i$  sends a request to which  $j$  has to reply with the last version of the mark. After receiving the mark from  $j$ ,  $i$  checks the signature and the timestamp and notifies node  $j$ . In the case when the test is failed, both the correct mark and the mark exchanged during this protocol can be sent to the authority by using GSM or broadcast in the network as a proof of the clone attack, and the credentials of node  $j$  are thus revoked. Of course, the legitimate node  $j$  is also invited to re-authenticate and get new credentials.



**Figure 2: Mark check protocol. Here  $CA$  denotes the trusted authority.**

Note that if a clone refuses to initiate or to complete either of the two protocols with the goal of avoiding detection, then it does not authenticate and its cloned certificates are going to expire soon thanks to Community Certificates, the other mechanism we propose in this paper that works very well when the adversary does not meet the members of the communities of the original owner.

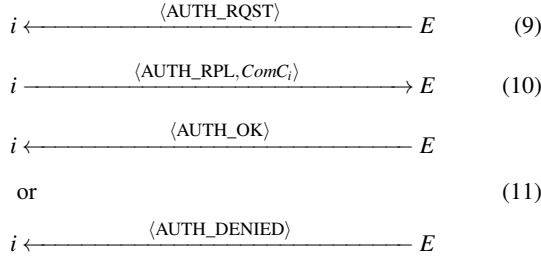
#### 4.1.2 Personal Marks, Insiders, and Outsiders

Personal Marks works on the assumption that both the clone and the original node get in touch with a common node in the community of the original node, sooner or later. This is often true—the adversary is someone that is part of the same community of the victim: A colleague at workplace, or a fellow student at college. In this case, we will say that the adversary is an *insider*, and Personal Marks works extremely well as shown later in the experiments with real mobility traces.

However, this is not always the case. It is perfectly possible that the clone uses the credentials that it has stolen far from the communities of the original node in the network. In this case, we will call the adversary an *outsider*. This is also a very important scenario, in many real cases the attack can be performed by someone that we don't know and that has the goal of taking our device to a different city or in a different part of the same city and use the credentials when needed. Clearly, Personal Marks can't do much against outsiders. Outsiders never meet someone from the community of the victim. Therefore, they are never checked and they don't need any fresh mark. To fight back outsiders, we have to develop a completely new mechanism.

## 4.2 Community Certificates

When a node, say node  $i$ , agrees to enter the system, it automatically enters a training period during which it securely collects signed and timestamped logs of the physical contacts with other nodes. At the end of the period the logs are reported to the authority. The authority uses the logs to build a signed certificate  $ComC_i$  that is sent back to node  $i$ . Clearly, all these messages are encrypted and authenticated, like all sessions in our protocols. The certificate is of the form  $ComC_i = \langle \langle FS_i, FI_i, k_i \rangle_{CA}, SU_i \rangle$ . In the certificate,  $FS_i$  is the set of “best friends” of node  $i$  (its community);  $FI_i$  is a mapping that tells the “strength of the relationship” between node  $i$  and his best friends. More specifically, for every  $j \in FS_i$ ,  $FI_i(j)$  is a value computed as a monotone function of the inter-contact times between nodes  $i$  and  $j$  observed during the training period. Moreover, for every  $j \in FS_i$ ,  $SU_i(j)$  is a timestamp signed by node  $j$  that certifies at what time there has been the last contact between node  $i$  and  $j$  (this object is similar to the mark in Personal Marks). Given node  $j \in FS_i$ , we say that the timestamp  $SU_i(j)$  is *fresh* if  $t < SU_i(j) + FI_i(j)$ . Certificate  $ComC_i = \langle \langle FS_i, FI_i, k \rangle_{CA}, SU_i \rangle$  is



**Figure 3: Community-Authentication protocol: Node  $i$  authenticates to entity  $E$  (either a node or a resource like an ATM machine/card reader etc.).**

valid at time  $t$  if and only if at least  $k$  signed timestamps in set  $SU_i$  are fresh. In other words, the certificate is valid only if the node has been able to collect enough fresh signatures by physically meeting people in his circle of friends. When the authority generates the certificate,  $SU_i$  can be prepared with timestamps signed by the authority, just to make the certificate immediately valid.

Note that all this process is totally automatic. Note as well that we assume that no attack occurs during the training period. This is quite reasonable indeed, if such an attack takes place, the authority would receive training logs from both the victim and the clone at the same time, thus revealing the attack.

#### 4.2.1 Community Certificates and Outsiders

The community certificate is a requirement for the authentication of node  $i$  to the other peers of the system. Indeed, when  $i$  meets another node, say node  $j$ , it is required to show the certificate  $\text{Com}C_i$  in case it is willing to set up a communication or to use one of the services provided by node  $j$ . The simple protocol is described in Figure 3.

As mentioned, an adversary that has the goal of using the cloned smart-phone outside the communities of the legitimate owner is an outsider. Community Certificates proves to be an excellent mechanism to stop clone attacks by outsiders. Indeed, when taken outside the communities of the owner, the certificate expires soon. The adversary is simply not able to collect enough fresh signatures to keep the certificate valid.

In the experiments with real traces, we show that it is possible to choose  $FS_i$ ,  $FI_i$ , and  $k$  in such a way that the certificates are continuously and consistently valid, when carried by the legitimate owner, and expire quickly when carried by outsiders. Therefore, a combined use of Community Certificates and Personal Marks is an excellent way to provide efficient and secure authentication and to thwart the clone attack in mobile wireless networks of smart-phones. In Section 6 we discuss a few ideas, add-ons, and solutions to extend the features of Community Certificates and to deal with non-frequent scenarios like the case when the legitimate owner has to use his own community certificate during a long travel far from his usual communities.

#### 4.2.2 Dynamic Community Certificates

So far, we have described Community Certificates as a static system. However, in real life it may be possible, even if it is not common, that we change our own community. In general, it is reasonable to imagine the following scenarios: (i) our community changes completely since, for example, we move to another town; and (ii) one of our friends moves away, or a new node is our new best friend. Here, we see that it is easy to design protocols to dy-

namically change the community certificate in a secure way.

In case (i), it is enough to start off a new training phase and to get a new certificate. In case (ii), we can initiate a selective update of the certificate to remove one node, or to add a new one, or to update the parameter of a node that is already part of our ring. Of course, the addition and/or the removal can change all the parameters of the certificate, like mapping  $FI_i$  or  $k_i$ . The procedure can be easily secured. Indeed, when the procedure starts, the authority sends a GSM message to the node. If a clone requests the procedure to change the certificate according to his own communities, than the message is received by the original owner as well, that promptly detects the attack and sends to the authority a signed request of certificate revocation.

### 4.3 Multiple, Coordinated Clone Attacks

In this paper, we consider the problem of detecting a single clone attack. Generally speaking, if the adversary is very powerful, it is however possible that it clones a whole set of mobile nodes. While we don't explicitly deal with this case in this work, it is still useful to see what is going to happen with Personal Marks and Community Certificates.

What we can say is that Personal Marks keeps working with (approximately) the same performance. According to our experiments, communities are not closed sets of nodes. Usually nodes belong to many different communities and the structure of the network is characterized by many *overlapping* communities. This is in accordance with other well-known research work in the literature [34]. In other words, it is usually impossible to select a small community of nodes that have very little contacts with nodes outside the community (of course, if you can select the whole network than it is easy, but if the adversary is able to clone the whole network that there is little we can do about it). Therefore, Personal Marks is still efficient in detecting the attack, thanks to all the interactions between the nodes of the cloned community with nodes outside the community.

A similar argument is true for Community Certificates—it is usually impossible to isolate a small community whose nodes do not have nodes that are outside the community in the certificate. However, it is in principle possible to isolate and clone a community in such a way that the cloned devices can authenticate each other. To make this attack possible only if the community to be cloned is very big, parameter  $k$  must be set high enough, and certificates should be computed in such a way not to form cliques. While this is out of the scope of this paper, it is certainly a open and interesting research direction. In any case, that means that the adversary is able to hijack a whole community. Moreover, we will see how to extend Community Certificate by admitting fixed devices in the certificate. For example, a fixed infrastructure in both our home and our office. While cloning a device can be easy, stealing a fixed infrastructure can be considerably harder.

## 5. EXPERIMENTAL RESULTS

In this section, we present some experimental results in order to show the detection performances of our system. The results refer to the two typologies of attack we consider: Insiders and outsiders. In the first case the detection relies on the Personal Marks sub-system. Rather, in the second, the Community Certificates sub-system comes into play and the detection is determined by the expiring time of the certificate when in the hand of the adversary. We start off by giving detailed information on the dataset we used to evaluate our protocol. Then we present the results obtained with Personal Marks and with Community Certificates.

**Table 1: Details on the datasets (DS) and respective training period (TR).**

Data set	Dartmouth	UCSD	Reality	SWIM
Total nodes	1101	32	45	1500
DS AVG active/day	1034	27	37	1500
TR AVG active/day	980	28	38	1500
DS AVG contacts/node/day	283	49	15	132
TR AVG contacts/node/day	263	51	16	131
DS AVG stability/node/week	0.55	0.52	0.55	0.49

## 5.1 Datasets

All the experiments have been performed by using a simulator driven by three type of traces: WLAN (real), bluetooth (real), and simulated traces. For the WLAN traces we used the *Dartmouth* [26] and *UCSD* [29] datasets. The bluetooth trace we used is the *Reality* [17], collected during the Reality Mining project at the MIT Media Lab. Finally, the simulated trace has been generated using the *SWIM* [31] model.

In contrast with the simulated scenario, in which we have been able to select the number of nodes and the length of the experiment, in the case of WLAN and bluetooth traces we had to select a period of time reasonably long that contained a set of nodes that are fairly active. The reason is that real traces often suffer from data loss and contain nodes whose activity is recorded only for a portion of the total time span covered by the trace. This might cause loss of performance of our system.

### 5.1.1 Dartmouth

The Dartmouth trace consists of the SMNP logs of the access points across the Dartmouth College campus from April 2001 to June 2004. To infer the contacts between the nodes we follow the assumption widely used in the literature that two nodes are able to communicate to each other whenever they are associated to the same access point ([30] [8]). From this trace we have selected a time span of 8 weeks, from January 5, 2004 to March 1, 2004, during which a set of 1101 nodes have recorded to have at least 50 contacts a day for at least the 80% of the days. This ensures us that these nodes stay active during the whole period we used to evaluate our protocol.

### 5.1.2 UCSD

The UCSD trace is part of the Wireless Topology Discover project (WTD) [29]. The trace consists of the logs extracted from PDA carried by a set of about 275 freshmen students of the University of California, in San Diego. The trace spans a period of 11 weeks between September 22, 2002 and December 8, 2002, during which each PDA periodically recorded the signal strength of all the APs in its range. To infer the contacts between the nodes we again used the assumption that two nodes can communicate as long as they are in the range of a the same access point. As also reported in [30], we have found that the trace is characterized by a steady decline of the user population that especially affects the last two weeks. For this reason we decided to restrict our tests to the first 8 weeks and to use only the set of 32 nodes that recorded at least 1 contact a day for at least the 80% of the days.

### 5.1.3 Reality

As opposed to the Dartmouth and UCSD traces that used WiFi radio logs, the Reality trace has been collected using short ranged bluetooth radios. More in detail, the trace included the bluetooth records collected by 94 cellphones distributed to student and fac-

ulty on the MIT campus during 9 months (from September 2004 to June 2005). We have chosen this particular trace since it is one of the few existing traces which encompasses a large number of nodes communicating through bluetooth technology for a long period of time. Like other traces, however, many nodes recorded very few to no sightings for long periods of time. In order to keep the amount of nodes high we thus restricted ourselves to a period of 8 weeks going from October 18th to December 13th 2004 and discarded the nodes that didn't record at least 1 contact a day for at least the 70% of the days. This selection yielded a final set of 45 nodes.

### 5.1.4 SWIM

This trace has been generated using the SWIM model ([31]) that has been shown to simulate well human mobility in conference and university campus environments. The SWIM simulator has also been shown [25] to be able to properly scale a reference scenario by keeping the nodes density constant. Thanks to this we have been able to replicate the statistical and social properties of the Cambridge Campus bluetooth data set [27] (that is only 11 days long) increasing both the number of nodes and the time span by keeping the same dynamics of the original real trace in terms of average number of contacts per user: The generated trace contains 1500 nodes and is 8 weeks long. Since the simulator keeps track of all the contacts between the nodes, it hasn't been necessary to preprocess the trace and we've been able to use the whole set of nodes.

## 5.2 Training Period

As we already discussed in the section 4.2, before a node can use Community Certificates, it is required to go through a *training period* during which it collects logs of contacts. At the same time, the Personal Marks system described in Section 4.1 assumes that each node know its community. For this reason we decided to split all the traces in two parts: the former used as a training period for all the nodes, the latter used to test the performances of our protocols. During the training period, we both collect data about the contacts between the nodes needed by the community certificates and extract the communities of the nodes. For this, we use the  $k$ -clique algorithm [34], one of the most popular in the area of social mobile networking ([20, 22, 21, 25]), of which there is a distributed version ([22]).

Since the movement of people is usually characterized by a high degree of regularity, we have seen that it is not necessary that the training period be long: In every trace we used a training period consisting of only the initial 25% of the total time span of the trace. This is enough for our system to quickly detect both insiders and outsiders. A further evidence that our intuition is good is given by the results shown in Table 1, where it can be seen that the properties of the trace during the training period are very close to those of the rest of the trace. In particular, the last row of the table shows what we have called the average *stability* of the nodes. We defined the stability of a node  $x$  as a measure of the average similarity between

the set of nodes  $x$  met during the training period and those it met during each week of the rest of the trace. More in detail, if  $T$  is the set of nodes that  $x$  met during the training period and  $S_i$  is the set of nodes that  $x$  met during the  $i$ -th week of the trace, the stability of  $x$  is computed as

$$\frac{1}{n} \sum_{i=1}^n \frac{|S_i \cap T|}{|T|}$$

where  $n$  is the number of weeks of the trace (training period excluded). As shown in the table, the average stability of the nodes is always close or greater than 0.5 meaning that, on average, a node meets every week at least half of the nodes it met during the training period.

### 5.3 Communities

In this section we briefly give details about the communities found in the various traces. To use the  $k$ -cliques community detection algorithm we first had to extract a social graph from the contacts happening in the training period of each trace. We put the edges in the graphs in such a way that they reflect the repetitive social behaviour of the nodes. Since all the not-simulated traces we used have been collected in university campuses, we expect the social connectivity to be given by students that, for example, frequent the same classes, study in the same library or live in the same dormitory. Once the social graph is built, the  $k$ -cliques community detection algorithm is run over it in order to find groups of nodes socially close to each other. However, it is important to note that according to the experiments our protocols are pretty robust with respect to different definitions of “friend” used in the literature, as long as it is true that friends meet regularly and frequently.

### 5.4 Personal Marks vs Insiders

To test the performance of Personal Marks, we run a trace-driven simulation where we make each pair of nodes of the same community check and exchange marks each time they meet.

In order to simulate the attack, for every node  $i$  we do the following: For every node  $j$  in the community of node  $i$  we generate a clone attack of node  $j$  against node  $i$  (node  $j$  is the insider) and measure the time required by Personal Marks to detect the attack. Performing the experiment is rather straightforward. As we described in Section 4.1, a node in the victim’s community detects the attack in two cases: The first case is when the node first meets the adversary and then the victim, the second case is when the node first meets the victim and then the adversary. In the first case, the mark of the victim node will be found to be outdated, while, in the second case, it is the mark of the adversary that will be found to be outdated. Thus, we measure the average time it takes for any of these two cases to happen after the attack happened.

Figure 4 shows the performance the Personal Marks sub-system on all the traces—real and synthetic ones. We show, for a given number of days  $x$ , what is the number of nodes for which the detection of the attack takes more than  $x$  days on average. First, we can notice from the results that, consistently in all the traces, the time needed to detect an insider is very low. Indeed, in the majority of the cases the attack is detected in one day, one day and a half at most. More in detail, in the Dartmouth trace (Figure 4(a)), the attack gets detected in less than one day in the 84% of the cases and in less than two days in the 96% of the cases. In the UCSD trace the attack is detected in less than one day in the 44% of the cases, but for the 97% of the cases it takes less than two days. In the Reality trace (Figure 4(c)) the performance of Personal Marks is slightly worse: In the 69.5% of the cases the attack is detected in less than one day and a half, while it takes less than three days for

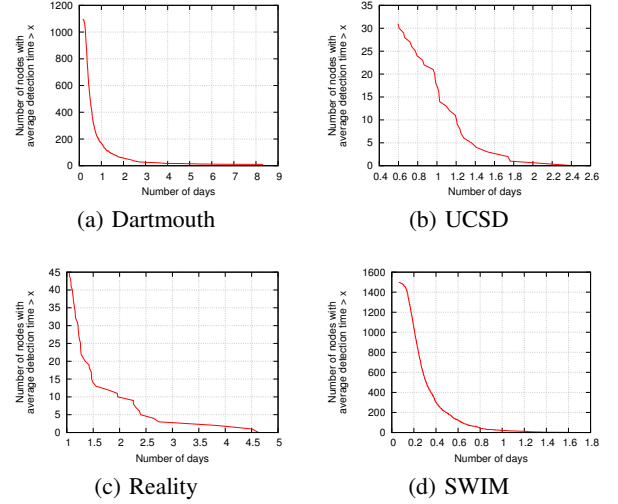


Figure 4: Distribution of the average detection time of insiders

the 93.4% of the cases. The reason why it takes more time to detect the attack is that the Reality trace is considerably sparser than the other traces. Indeed, Table 1 shows that the average number of contacts is the minimum among all the traces. Lastly, Personal Marks shows the best performances in the SWIM trace (Figure 4(d)): In the 86.5% of the cases, the nodes need at most half a day to detect the attack, while in the 99% of the cases it takes at most one day and a half. This is not surprising. In the real traces, in fact, all the nodes have periods of time (that can last even a few days) where they didn’t report any contact. Of course, this is not very realistic—all of the traces capture only a very small fraction of the real contacts of the user (the other people in the experiment, often a small subset of the other students at school). We expect that, in case of wide deployment in the real world, the number of contacts would be much higher than any of these traces—both real and synthetic ones—and the performance, already good, would greatly improved even compared to the synthetic traces, that have been designed to mimic the dynamics of the Cambridge traces.

### 5.5 Community Certificates vs Outsiders

The Community Certificate sub-system has the goal to fight back outsiders. This sub-system has three main components: Set  $FS_i$  of nodes appearing in the certificate of node  $i$ , the values in  $FI_i$ , and parameter  $k_i$ .

Let us concentrate on  $FS_i$  and  $FI_i$ . The first one is the set of nodes from which node  $i$  receives the signed timestamps it uses to prove its identity. The second one defines for how long a signed timestamp given to  $i$  by  $j$  is *fresh* (see Section 4.2). Clearly, the correct definition of  $FS_i$  and  $FI_i$  is crucial. For instance, if the set  $FS_i$  were large and the values  $FI_i(j)$  for every  $j$  in  $FS_i$  were big, then it would be easy for node  $i$  to prove its identity. In fact, the probability that node  $i$  meets at least  $k_i$  nodes in the set  $FS_i$  would be large, and, at the same time, the signed timestamps  $SU_i(j)$  that node  $i$  would receive from them would last for long time. This would surely translate into a low number of false positives in our system (legitimate users that cannot authenticate). At the same time, however, this would give more chances to the adversary to keep his certificate valid for a long period of time. Thus, by choosing the elements of set  $FS_i$  and the values of  $FI_i$  we trade-off the ability of the system to correctly authenticate honest nodes and the performance in



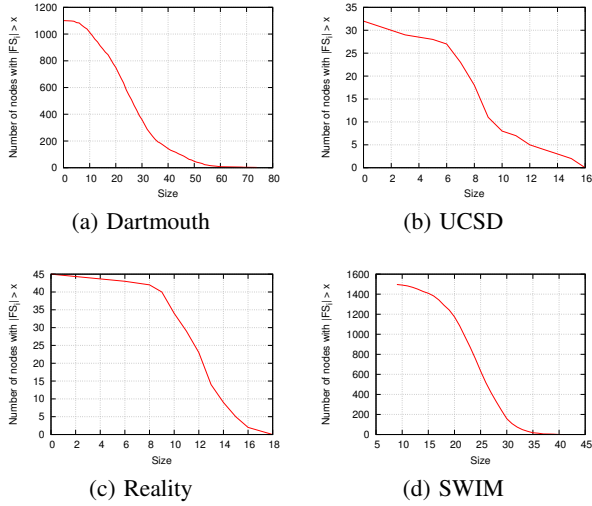


Figure 5: Distribution of the sizes of the sets  $FS_i$

blocking outsiders.

Given these considerations, an intuitive choice is that of putting in  $FS_i$  the set of nodes that  $i$  is likely to encounter more often, and in  $FI_i(j)$  the expected inter contact times between  $i$  and each of the nodes in  $FS_i$ . To define  $FS_i$  and  $FI_i$  we again use the social behavior of node  $i$  observed during the training period as an approximation of its behaviour the rest of the trace. To compute  $FS_i$  set, we first consider the set of nodes  $S_i$  that  $i$  has seen at least once during the training period. Then, for every node  $j \in S_i$  we calculate the number  $d_{ij}$  of days in which  $i$  met  $j$ , and compute the average value:

$$\bar{d}_i = \frac{1}{|S_i|} \sum_{j \in S_i} d_{ij}$$

Finally, we take as  $FS_i$  the set of nodes  $j$  such that  $d_{ij} \geq \bar{d}_i$ . In Figure 5 we show the distribution of the size of the sets  $FS_i$  in all the traces. For the Dartmouth trace (Figure 5(a)) the average size of  $FS_i$  is 26.4, for UCSD (Figure 5(b)) it is 9, for Reality (Figure 5(c)) it is 12.3 and for SWIM it is 24.3. It is worth to observe that the sizes of the  $FS_i$  sets don't change much from trace to trace despite the fact that in Dartmouth and SWIM the number of nodes is more than 20 times bigger than that of UCSD and Reality. We interpret this fact as an hint that our definition of  $FS_i$  is meaningful and rather stable.

The values in  $FI_i$  are computed starting from the inter-contacts observed between node  $i$  and the nodes in  $FS_i$  during the training period. More in detail, for every pair of nodes  $(i, j)$  we computed  $FI_i(j)$  as the average number of days between contacts of nodes  $i$  and  $j$ .

### 5.5.1 False Positives

First, we tested the system in order to know if the honest nodes are always able to certificate their identity. We define a false positive as the event that a legitimate user is not able to authenticate since he is temporarily out of his community for a long time. To perform these tests for all the nodes we have set the value  $k_i$  to 1 and run the simulation as if no attack happens. Setting  $k_i$  to 1 means that for a node it is sufficient to have just one valid signed update to prove its identity. This permits us to study the lower bound of the number of false positives the system can ensure given the defi-

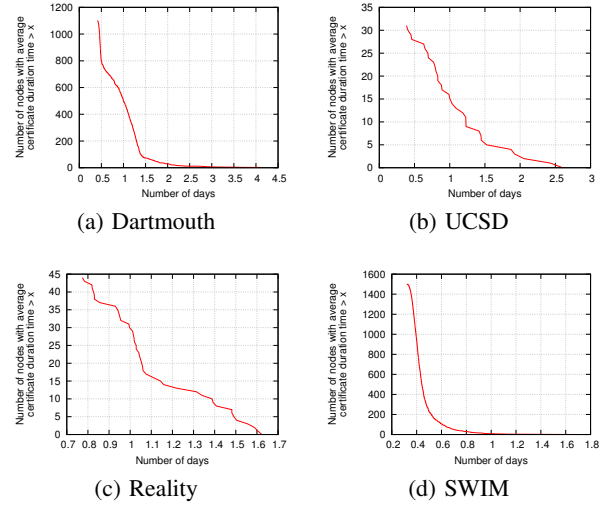


Figure 6: Distribution of the average community certificates durations

nitions of  $FS_i$  and  $FI_i$  we provided above.

In the case of the Dartmouth trace, we have found that 43 nodes (that is just the 4% of all the 1101 nodes) show false positives: 36 of them have only one false positive, while the rest have two false positives. Moreover, the periods in which these nodes are not able to prove their identity always lasts one day at most. Considering the simple way in which we defined the sets  $FS_i$  and the values  $FI_i$  these results are pretty good. In the case of the UCSD trace our tests performed even better: only one node over the 32 we tested (that is, the 3% of the nodes) has false positives. In particular the node is not able to prove its identity only for three time intervals, all of them lasting less than one day. We got a similar result in the Reality trace too: only one node over 45 (2% of the nodes) had false positives and this happened only twice and for less than one day. Finally, in the SWIM trace no node has generated false positives.

### 5.5.2 Certificates Duration

We have shown that the way we defined  $FS_i$  and  $FI_i$  is good enough to ensure that only a very small percentage of honest nodes suffers of false positives. Now it is time to study the effectiveness of the Community Certificates sub-system in limiting the power of a clone attack by an outsider. To do that, we study is the average time it takes for the community certificate of the nodes to expire in absence of contacts with nodes in the set  $FS_i$ . This gives us a measure of the amount of time the adversary is able to use a cloned identity. The reason is that, once the adversary cloned an identity, he/she won't be able to use it inside the community of the victim or to renew it without being quickly detected by the Personal Marks sub-system. In such a situation, the only option left to the adversary is that of using the identity far from the victim's community.

When we studied the average duration of the community certificates, we used, for each node, the highest value  $k_i$  for which the node is not affected by false positives. This gives us an idea of how much we can stress the system to get the lowest possible average certificate duration given our definitions of  $FS_i$  and  $FI_i$ . The results of our measurements are shown in Figure 6. Each graph shows, for a given number of days  $x$ , what is the number of nodes whose certificate has, on average, a duration of more than  $x$  days when cloned



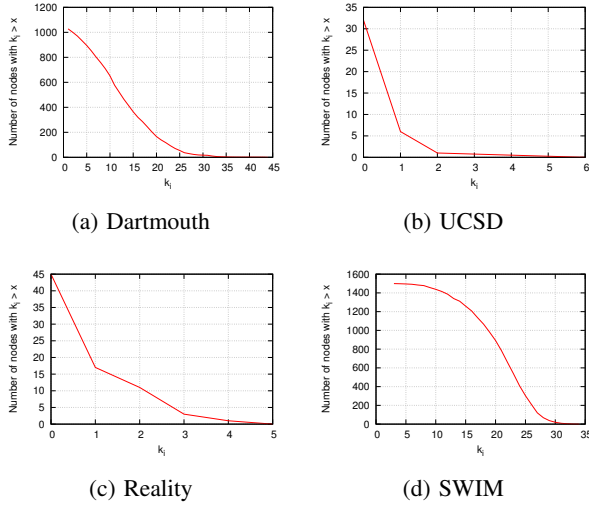


Figure 7: Distribution of the values  $k_i$

by an outsiders. So, the faster the better. The first thing we can notice is that in all the traces the average duration of the certificate is never more than 2 or 3 days in the worst cases, and roughly between 0.5 and 1.5 in the majority of the cases. More in detail, in the Dartmouth trace (Figure 6(a)) the community certificate has an average validity of less than one day and a half for 93% of the nodes, while only 3% of the nodes have a certificate lasting, on average, more than two days. In the UCSD trace (Figure 6(b)), the community certificates last on average at most one day and a half for 84% of the nodes. This value is slightly smaller than the corresponding one found in the Dartmouth trace. However, in the worst case the average certificate duration in the UCSD trace is 2.6 days as compared to the more than 3.5 days of the Dartmouth trace. In the case of the Reality trace, the average certificate duration is at most one day and a half for the 91% of the nodes and it is at most 1.62 days long for all the nodes. This value is the minimum among the maximum average certificate duration we measured in the three real traces we have used. Finally the simulated trace SWIM shows very good performance: 99% of the nodes have an average certificate duration smaller than 1 day and the maximum average certificate duration is 1.78 days.

Figure 7 shows the distributions of the values  $k_i$  that we have found to be the maximum we could use on each node without generating false positives. As we can see, there is a big difference between the optimal values found for the Dartmouth and SWIM traces (Figures 7(a) and 7(d)) and those found in the Reality and UCSD traces (Figures 7(c) and 7(b)). In the first two traces there is a big number of nodes for which we have been able to select (automatically from the training period) high values of  $k_i$  without producing false positives. For instance, in the Dartmouth trace, for 93% of the nodes we have been able to select a value  $k_i$  greater than 1 and for 52% of the nodes a value greater than 11. In SWIM, the sets  $FS_i$  are generally smaller than those found in Dartmouth (see Figure 5) and this translates into smaller values of  $k_i$ . Nevertheless in SWIM it is possible to set a value  $k_i$  greater than 1 for 99% of the nodes and a value greater than 10 for 42% of the nodes. On the other hand, in UCSD and Reality we can use a  $k_i$  greater than one only for 19% and for the 37.7% of the nodes respectively. This big difference can be explained by the fact that, as shown in Table

1, in Dartmouth and SWIM the nodes have a much higher number of contacts than in UCSD and Reality. In the case of Dartmouth this happens because the number of nodes is high and the trace has been collected using WiFi data rather than a short ranged bluetooth radio. On the other hand, in SWIM, since we are simulating a bluetooth trace, the nodes don't have a range as high as that of Dartmouth hence they tend to have contacts with smaller portions of the network. However, as already stressed in the previous sections, using the SWIM simulator we never miss a contact between two nodes and, moreover, we don't have data losses typical of the real traces.

Overall, we can say that the definitions we used for  $FS_i$  and  $FS_j$  seem to work well in different scenarios. Indeed, they make it easy to the large majority of the honest nodes to keep their community certificates up to date. At the same time, the adversary that clones an honest identity, will be able to use it for only short period of time that is in the order of one or two days at most.

## 5.6 Using Fixed Infrastructure in the Community Certificates

So far we have assumed that each node receives its signed timestamps from the other nodes in its community. However, the Community Certificates protocol is actually more general than that. Indeed, the certificate authority may potentially put into the set  $FS_i$  the ID of any kind of device with a private/public key pair that is able to send signed timestamps to the nodes that enter its range of communication. For instance, we could think of a WiFi access point in a university campus. In such a scenario it would be perfectly reasonable for a student to have stored in the set  $FS_i$  of his/her smartphone the public key of the access points of the university buildings he/she frequents the most. There would be three good reasons to do that. The first one is that there are locations, just like friends, that we "meet" more often than others; it is easy to check experimentally that the pattern of meetings is similar to that of two nodes carried by humans. The second one is that the access points have fixed locations and are, usually, always on. The third one is that it is supposedly harder to tamper with an access point and to move it to another location rather than a mobile device. This would make it more difficult for the adversary to refresh the community certificate of a cloned identity  $i$  by compromising a given number of nodes in the set  $FS_i$  (see Section 4.3).

For these reasons we found it interesting to test the performance of Community Certificates in a situation where the certificate authority puts in  $FS_i$  the access points of the most frequented locations of each node. To do that, we used the Dartmouth and UCSD traces since they have been collected exactly recording the associations between mobile nodes and the access point of the Dartmouth and University of California San Diego campuses respectively (see Section 5.1). We performed the tests with the same definitions of  $FS_i$  and  $FI_i$  we used in Section 5.5, though, this time, we are interested in the contacts between nodes and access points rather than between pair of nodes. The results are shown in Figures 8 and 9.

The first difference we can notice between these results and the previous ones is the change in the sizes of the sets  $FS_i$ . In the case of the Dartmouth trace (Figure 8(a)) the average size of  $FS_i$  is now only 3, while before it was 26. On the contrary, in the UCSD (Figure 9(a)) trace the size of the  $FS_i$  increased from an average of 9 to an average of 18. This is a consequence of the fact that in Dartmouth the contacts between the nodes happened in a restricted set of places, while, in UCSD, the nodes were generally more mobile. Note that these values are small as compared to the total number of access points, that is about 500 in both traces. The second difference is that the average certificate duration of the nodes slightly

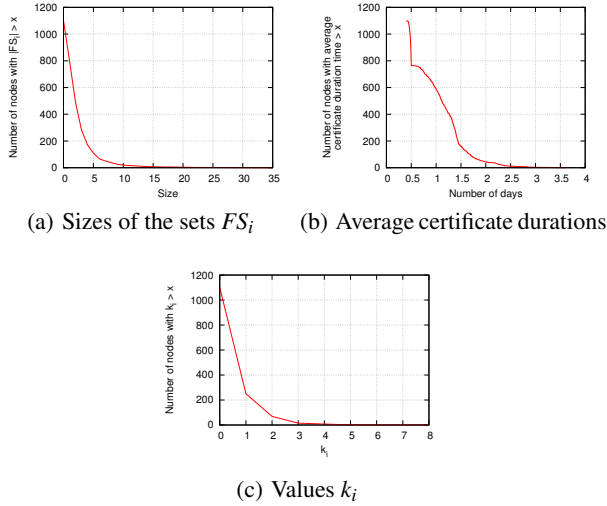


Figure 8: Dartmouth (Using Access Points)

increases with respect to our previous results (remember, however, that in these experiments we have removed the mobile friends!). In the case of Dartmouth (Figure 8(b)), the percentage of nodes whose community certificate has an average validity of less than one day has dropped from 54% to 46%. Also, the community certificates have an average duration of less than 1.5 days for 85% of the nodes while before it was for 93%. In the case of UCSD (Figure 9(b)) 31% of the nodes have a community certificate lasting on average less than one day (before it was 53%) while for 81% it lasts less than one day and a half (was 84%). To summarize, the results are not as good as with real mobile friends. However, they are good enough to make it reasonable to complement the set  $FS_i$  with a small number of entities that are part of the fixed infrastructure. This is not going to reduce the performance considerably and helps build a stronger system against coordinate attacks to multiple mobile nodes. Lastly, Figures 8(b) and 9(b) show distribution of the values of  $k_i$ .

## 6. ADD-ONS AND CAVEATS

Community Certificates can be complemented and extended in such a way to design more sophisticated and complete versions. Here we discuss a few of these add-ons.

### Travelling.

If we travel, especially alone, our community certificate is going to expire soon. While this is generally true, we can easily design a few solutions: (i) The user can suspend the community checks on the certificate during the travel; (ii) if we often travel between two cities, like Rome and San Diego, the user can prepare two different *profiles*, in the same certificate, that can be selected when changing community. Of course, it is fundamental that these solutions can be used only by the legitimate owner by performing the burdensome alternative authentication that is used in case of exceptional events.

### Complex patterns.

It is fairly easy to extend Community Certificates in such a way that it is not enough to get  $k$  fresh signed timestamps from your circle of friends. Rather, the certificate needs  $k_1$  fresh signed times-

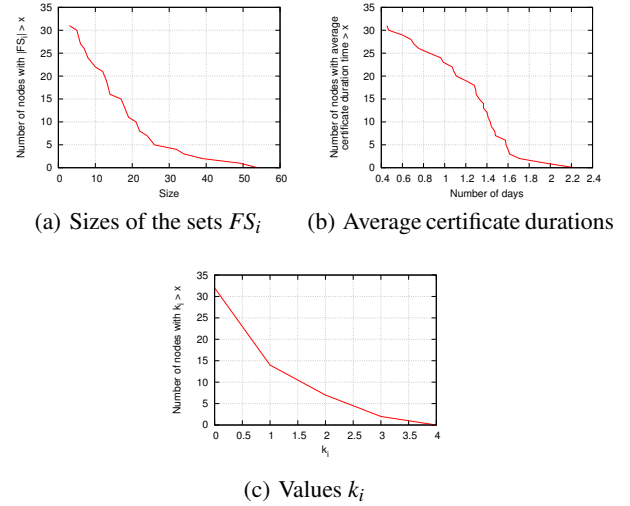


Figure 9: UCSD (Using Access Points)

tamps from one sub-community (for example your family) and  $k_2$  fresh signed timestamps from another sub-community (for example your colleagues) to be valid. A community certificate like this can easily be more efficient and more useful in particular situations. For example when we need at least one signature from one of our bosses to be able to perform critical operations. Or at least one signature from a fixed infrastructure in the building of our company, for example.

### Electronic handshakes.

In addition to standard physical contacts, Community Certificates can be extended to use *electronic handshakes*. Assume that, when meeting a friend, you use a particular secure "handshake" made with your devices (like bumping the two together, for example). Handshakes can be used at the place of physical proximity. In this way, contacts are more trustworthy, but, on the other hand, they need human intervention and are much fewer in number. Alternatively, handshakes could be used to exchange a stronger signed timestamp. By using *complex patterns*, one or more of these handshakes can be required to enable a higher level of authentication to perform critical operations.

## 7. CONCLUSIONS

In this paper we introduce Personal Marks and Community Certificates. The fundamental idea of Community Certificates is that, in networks of mobile people, authentication can be based on the notion of community, and nodes can authenticate by showing that they indeed meet the people that are part of their community. While the social structure of these networks has been extensively used in networking, to the best of our knowledge this is the first time that this has been used as a biometric to authenticate device. We also present Personal Marks, a way a community can use to protect itself against insiders performing a clone attack. The combined use of these mechanisms deliver an excellent protection of the social mobile network against the clone attack. Indeed our experiments show that the detection is always correct, and fast enough considered the slow dynamics of the trace we have used. In any real system, we believe that the much faster dynamics of meetings can help deliver much faster detection times.

## 8. REFERENCES

- [1] M. Barbeau, J. Hall, and E. Kranakis. Detecting impersonation attacks in future wireless and mobile networks. In *Proceedings of MADNES 2005 - Workshop on Secure Mobile Ad-hoc Networks and Sensors*. SVLNCS, 2005.
- [2] R. Bolle, S. Pankanti, and A. K. Jain. *Biometrics, Personal Identification in Networked Society: Personal Identification in Networked Society*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
- [3] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *Proceedings Eurocrypt*, 1993.
- [4] R. Brooks, P. Govindaraju, M. Pirretti, N. Vijaykrishnan, and M. Kandemir. On the detection of clones in sensor networks using random key predistribution. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(6):1246–1258, 2007.
- [5] H. Cai and D. Y. Eun. Crossing over the bounded domain: from exponential to power-law inter-meeting time in manet. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 159–170, New York, NY, USA, 2007. ACM.
- [6] H. Cai and D. Y. Eun. Toward stochastic anatomy of inter-meeting time distribution under general mobility models. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 273–282, New York, NY, USA, 2008. ACM.
- [7] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Pocket switched networks: Real-world mobility and its consequences for opportunistic forwarding. Technical report, Computer Laboratory, University of Cambridge, 2006.
- [8] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on opportunistic forwarding algorithms. *IEEE Transactions on Mobile Computing*, 6:606–620, June 2007.
- [9] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, pages 197–, Washington, DC, USA, 2003. IEEE Computer Society.
- [10] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:15:1–15:58, July 2009.
- [11] H. Choi, S. Zhu, and T. L. Porta. Set: detecting node clones in sensor networks. In *SecureComm07: IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2007.
- [12] M. Conti, R. Di Pietro, A. Gabrielli, L. V. Mancini, and A. Mei. The smallville effect: social ties make mobile networks more secure against node capture attack. In *Proceedings of the 8th ACM international workshop on Mobility management and wireless access*, MobiWac '10, pages 99–106, New York, NY, USA, 2010. ACM.
- [13] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei. Distributed detection of clone attacks in wireless sensor networks. *IEEE Transactions on Dependable and Secure Computing (IEEE)*, (to appear).
- [14] M. Conti, R. Di Pietro, L. V. Mancini, and A. Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, pages 80–89, New York, NY, USA, 2007. ACM.
- [15] E. M. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '07, pages 32–40, New York, NY, USA, 2007. ACM.
- [16] J. R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, 2002. Springer-Verlag.
- [17] N. Eagle and A. S. Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from <http://crawdad.cs.dartmouth.edu/mit/reality>, July 2005.
- [18] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '08, pages 251–260, New York, NY, USA, 2008. ACM.
- [19] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 41–47, New York, NY, USA, 2002. ACM.
- [20] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot. Pocket switched networks and human mobility in conference environments. In *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, WDTN '05, pages 244–251, New York, NY, USA, 2005. ACM.
- [21] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Mobile Ad Hoc Networking and Computing*, pages 241–250, 2008.
- [22] P. Hui, E. Yoneki, S. Y. Chan, and J. Crowcroft. Distributed community detection in delay tolerant networks. In *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*, MobiArch '07, pages 7:1–7:8, New York, NY, USA, 2007. ACM.
- [23] N. James, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, IPSN '04, pages 259–268, New York, NY, USA, 2004. ACM.
- [24] T. Karagiannis, J.-Y. Le Boudec, and M. Vojnović. Power law and exponential decay of inter contact times between mobile devices. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, MobiCom '07, pages 183–194, New York, NY, USA, 2007. ACM.
- [25] S. Kosta, A. Mei, and J. Stefa. Small world in motion (SWIM): Modeling communities in ad-hoc mobile networking. In *Proceedings of The 7th IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2010)*, Boston, MA, U.S.A., June 2010.
- [26] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo. CRAWDAD data set dartmouth/campus (v. 2009-09-09). Downloaded from <http://crawdad.cs.dartmouth.edu/dartmouth/campus>, Sept. 2009.
- [27] J. Leguay, A. Lindgren, J. Scott, T. Riedman, J. Crowcroft, and P. Hui. CRAWDAD trace upmc/content/imote/cambridge (v. 2006–11–17). Downloaded from <http://crawdad.cs.dartmouth.edu/-upmc/content/imote/cambridge>, November

2006.

- [28] Q. Li, S. Zhu, and G. Cao. Routing in socially selfish delay tolerant networks. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 857–865, Piscataway, NJ, USA, 2010. IEEE Press.
- [29] M. McNett. Wireless topology discovery. <http://my.url.com/>, 2008.
- [30] M. McNett and G. M. Voelker. Access and mobility of wireless pda users. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9:40–55, April 2005.
- [31] A. Mei and J. Stefa. Swim: A simple model to generate small mobile worlds. In *INFOCOM'09*, pages 2106–2113, 2009.
- [32] A. Mei and J. Stefa. Give2get: Forwarding in social mobile wireless networks of selfish individuals. In *ICDCS*, 2010.
- [33] Mobile payment in china. <http://www.mobilepaymentchina.com/mobilepayment/>.
- [34] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, June 2005.
- [35] B. Parno, A. Perrig, and V. Gligor. Distributed detection of node replication attacks in sensor networks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, pages 49–63, Washington, DC, USA, 2005. IEEE Computer Society.
- [36] Phone cloning. [http://en.wikipedia.org/wiki/Phone\\_cloning](http://en.wikipedia.org/wiki/Phone_cloning).
- [37] R. Singh, P. Bhargava, and S. Kain. Cell phone cloning: a perspective on gsm security. *Ubiquity*, 2007:1:1–1:8, July 2007.
- [38] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, pages 41–50, Washington, DC, USA, 2004. IEEE Computer Society.
- [39] S. Čapkun, L. Buttyán, and J.-P. Hubaux. SECTOR: secure tracking of node encounters in multi-hop wireless networks. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, SASN '03, pages 21–32, New York, NY, USA, 2003. ACM.
- [40] B. Zhu, S. Setia, S. Jajodia, S. Roy, and L. Wang. Localized multicast: Efficient and distributed replica detection in large-scale sensor networks. *IEEE Transactions on Mobile Computing*, 9:913–926, July 2010.